

Ngspice: an Open Platform for Modeling and Simulation from Device to Board Level

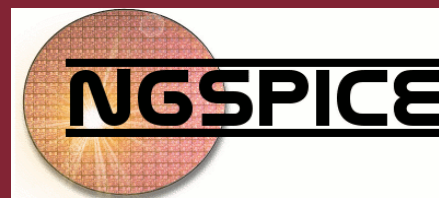
Paolo Nenzi^{1,2}, Vittorio Delitala², Marco Garzuoli², Robert Larice¹, Antonio Mastrandrea², Mauro Olivieri², Stefano Perticaroli², Fabrizio Ramundo², Lionel Sainte-Cluque¹, Ljiljana Trajkovic³, Holger Vogt², Dietmar Warning²

1) Ngspice development team; 2) DIET, “Sapienza” – Università di Roma; 3) School of Engineering Science, Simon Fraser University



SAPIENZA
UNIVERSITÀ DI ROMA

MOS-AK/GSA Workshop, December 8th 2010
San Francisco (CA)



Agenda

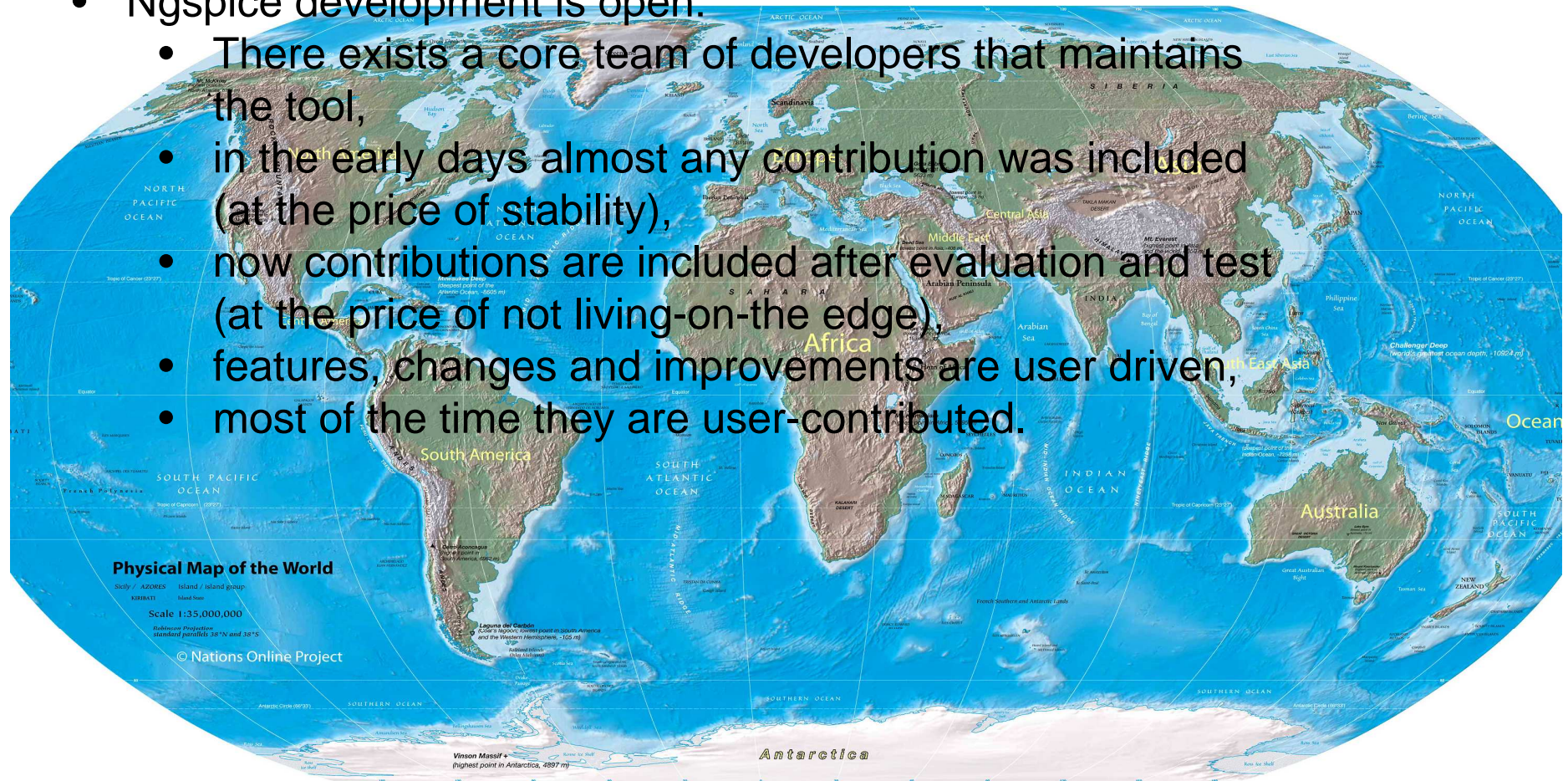
- Back home... 40 years later
 - Ngspice improvements over spice3
 - Ngspice as simulation kernel in “multi-domain” simulations
 - TCLspice and device-level simulation.
- Applications of ngspice at DIET (Uni Sapienza)
 - Ngspice profiling
 - Statistical delay modeling of nano- CMOS standard cells
 - Static leakage estimation in logic cells
 - BTBT leakage introduction into BSIM4
- Future developments
 - Steady state analysis
 - HomSpice and HomSSpice integration
- Conclusions

A spicy history

- '70s:
 - **1970** L. Nagel, R. Rorher: CANCER (Computer analysis of Non-Linear Circuits Excluding Radiation).
 - **1972** L. Nagel, D. Pederson: SPICE1 (Simulation Program with Integrated Circuit Emphasis).
 - **1975** E. Cohen, L.Nagel: SPICE2.
- '80s:
 - **1983** A. Vladimirescu, Zhang, Newton, D. Pederson, A. Sangiovanni-Vincentelli: SPICE2G.6, latest release of SPICE2.
 - 1989 T. Quarles: Development of SPICE3 starts.
- '90s:
 - **1993** T. Quarles, SPICE3f
 - **1997** T. Quarles, SPICE3f5
 - **1999** NGSPICE

Ngspice development model

- Ngspice is an open source software
- Ngspice development is open:
 - There exists a core team of developers that maintains the tool,
 - in the early days almost any contribution was included (at the price of stability),
 - now contributions are included after evaluation and test (at the price of not living-on-the edge),
 - features, changes and improvements are user driven,
 - most of the time they are user-contributed.



Improvements in ngspice

Devices

- Expression defined
 - resistance,
 - capacitance,
 - inductance.
- R,L or C can be a (non-linear) function of:
 - node voltages,
 - branch currents.

Example:

* Non-linear resistor

```
.param R0=1k Vi=1 Vt=0.5
```

* res. depending on control voltage V(rr)

```
R1 rr 0 r='V(rr) < {Vt} ? {R0} : {2*R0}'
```

* control voltage

```
V1 rr 0 PWL(0 0 100u {Vi})
```

```
.control
```

```
set noaskquit
```

```
tran 100n 100u uic
```

```
plot i(V1)
```

```
.endc
```

```
.end
```

Improvements in ngspice

Devices (2)

- Improved “B” arbitrary source:
 - Added *time*, *temper* and *hertz* variables,
 - added ternary operator (a?b:c),
 - piecewise linear function source.
- Nonlinear VCCS and VCVS sources

Example:

* PWL arbitrary source

**Bdio 1 0 I = pwl(v(A), 0,0, 33,10m,
+ 100,33m, 200,50m)**

* Non linear VCVS (E element)

E41 4 0 vol = 'V(3)*V(3)-Offs'

* Non linear VCcS (G element)

G51 55 225 cur = 'V(3)*V(3)-Offs'

Improvements in ngspice

Devices (3)

- Transmission lines:
 - Bug fixes and corrections on existing code
- KSPICE models:
 - Single Lossy Transmission Line (TXL)
 - Coupled Multiconductor Line (CPL)
 - Ngspice supports up to 8 coupled lines

Example:

* TXL example

Y1 1 0 2 0 ymod LEN=2

```
.MODEL ymod txl R=12.45 L=8.972e-9 G=0  
+ C=0.468e-12 length=16
```

* CPL example

P1 in1 in2 0 b1 b2 0 PLINE

```
.model PLINE CPL length={Len}  
+ R=1 0 1 L={L11} {L12} {L22} G=0 0 0  
+ C={C11} {C12} {C22}  
.param Len=1 Rs=0  
+ C11=9.143579E-11 C12=-9.78265E-12  
+ C22=9.143578E-11 L11=3.83572E-7  
+ L12=8.26253E-8 L22=3.83572E-7
```

Improvements in ngspice

Semiconductor devices (non MOS)

- Almost all device models have been modified to correct bugs.
- Some models (HFET and MES 2,3,4) comes from modeling books.

Type	Level	models
D	1	Original spice diode model
BJT	1	Original spice gummel-poon (SGP)
<i>BJT</i>	<i>2</i>	<i>Improved SGP model for lateral and vertical devices</i>
<i>BJT</i>	<i>4</i>	<i>VBIC model</i>
JFET	1	Original spice model
<i>JFET</i>	<i>2</i>	<i>Parker-Skellern model</i>
MES	1	Original Statz spice model
<i>MES</i>	<i>2,3,4</i>	<i>Models by Ytterdal</i>
<i>MES</i>	<i>5,6</i>	<i>HFET models</i>

Improvements in ngspice

Semiconductor devices (MOS)

- Ngspice implements most recent MOS models that have been made available for free
- EKV model has two implementations:
 - One “long-channel” 2.6 version is available in the standard distribution
 - The full C-code implementation is available for free through NDA.
- Some old BSIM3 implementations are retained for compatibility purposes

Type	Level	models
MOS	1	Spice3 level 1 model (“Shichman-Hodges”)
MOS	2	Spice3 level 2 model (“Grove Frhoman”)
MOS	3	Spice3 level 3 model
MOS	4	BSIM1
MOS	5	BSIM2
MOS	6	Spice3 level 6 model
MOS	8	BSIM3 version 3.3.0 model
MOS	9	<i>MOS9 model (implementation by Alan Gillespie)</i>
MOS	10	<i>BSIM4 SOI model version 4.3.1</i>
MOS	14	<i>BSIM4 model version 4.6.5</i>
MOS	17	<i>HISIM1 model</i>
MOS	29	<i>B3SOI-PD model</i>
MOS	30	<i>B3SOI-FD model</i>
MOS	31	<i>B3SOI-DD model</i>
MOS	44	<i>EKV model</i>
MOS	49	<i>BSIM3V1S model (Serban Popescu)</i>
MOS	50	<i>BSIM3V1 model</i>
MOS	51	<i>BSIM3V1A model (Alan Gillespie)</i>
MOS	52	<i>BSIM3V0 model</i>
MOS	62	<i>STAG model</i>

Improvements in ngspice

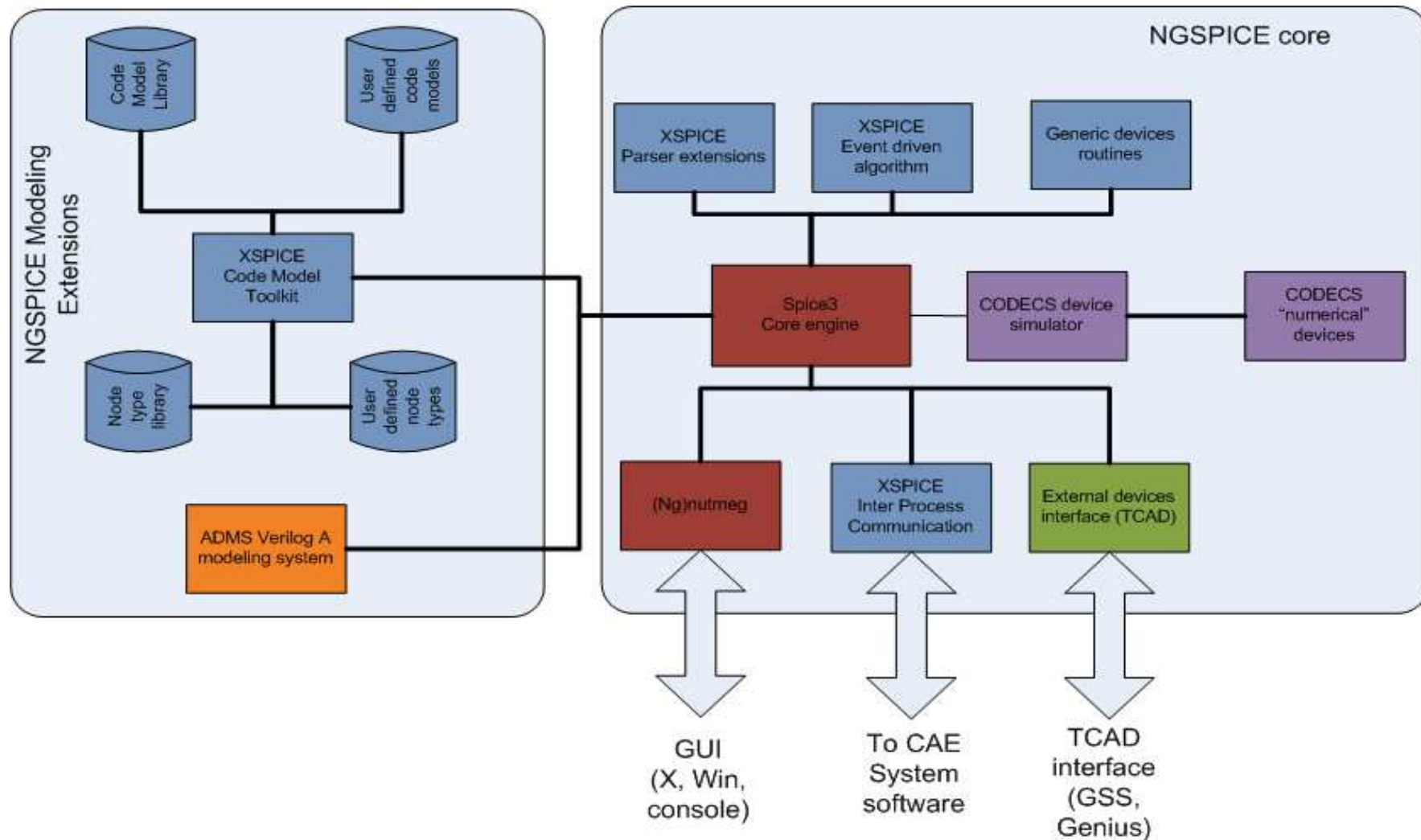
Frontend improvements

- Leakage:
 - Original spice3f5 had severe memory allocation problems that prevented multiple simulations without exiting from the software.
 - Most of the condition that caused ngspice to segfaults have been corrected.
- Parametric netlists:
 - Ngspice supports the use of parameters and user defined functions in netlists.
- MEAS(ure):
 - This command analyze the output of DC, AC and transient analysis and reports quantities as propagation delay, rise and fall times, peak to peak voltage, etc.

- OpenMP implementation:
 - BSIM3 and BSIM4 models can use OpenMP to exploit parallelism of multicore – multiprocessor machines.

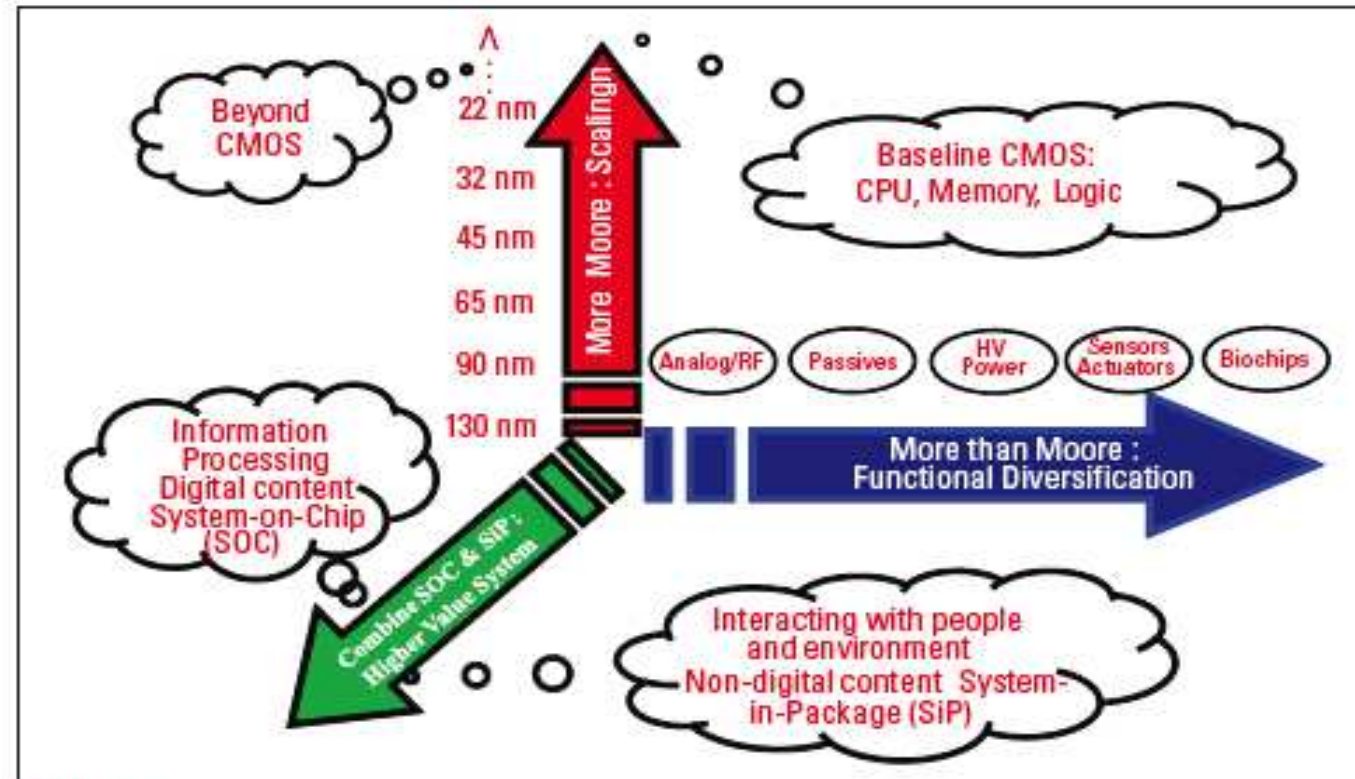
Ngspice as simulator

simply another spice...



Re-thinking spice and ngspice

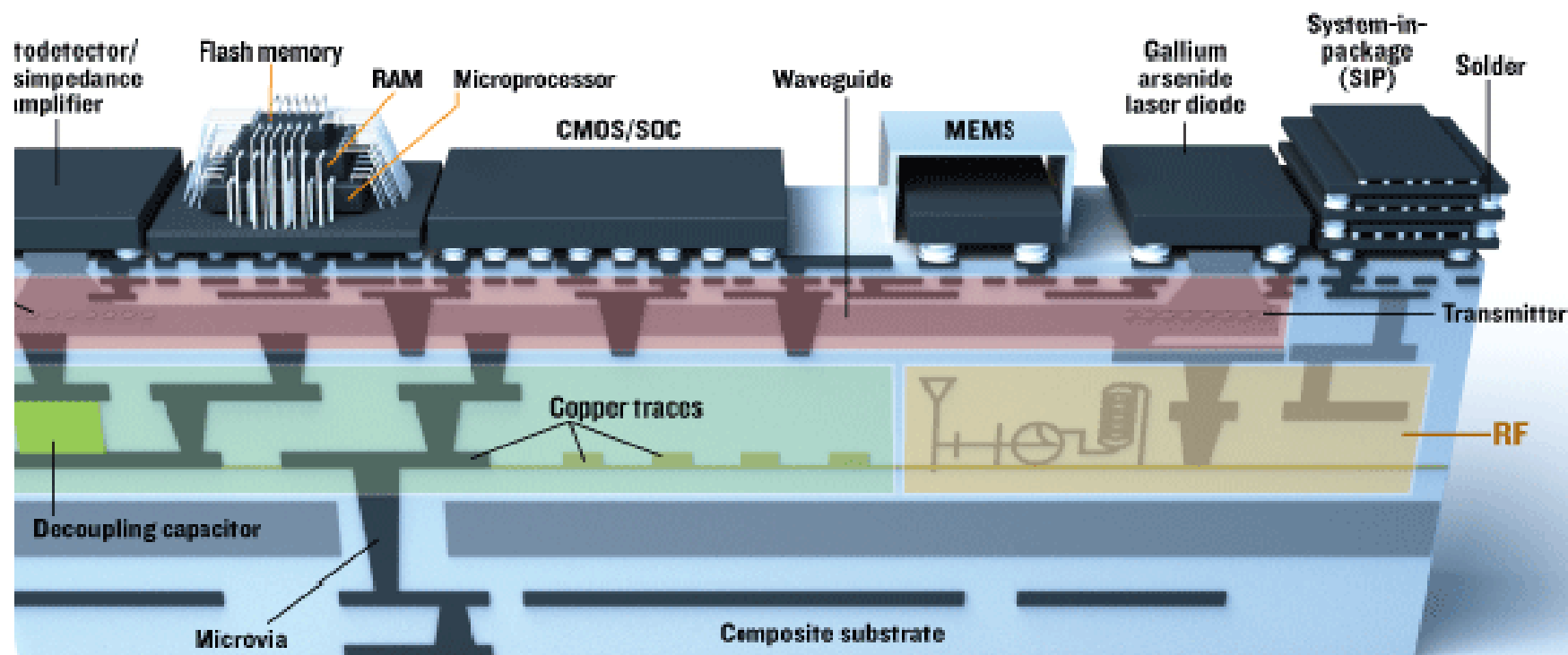
- Deep sub-micron devices are described by complex analog models.
- Their intrinsically analog behavior cannot be overlooked in digital IC design verification.
- Modern SOC integrate different functions (logic, analog, memory, MEMS, power management)
- Design verification process involves different domains (electrical, DSP, transactions).



Source: ITRS

Can spice manage this complexity ?

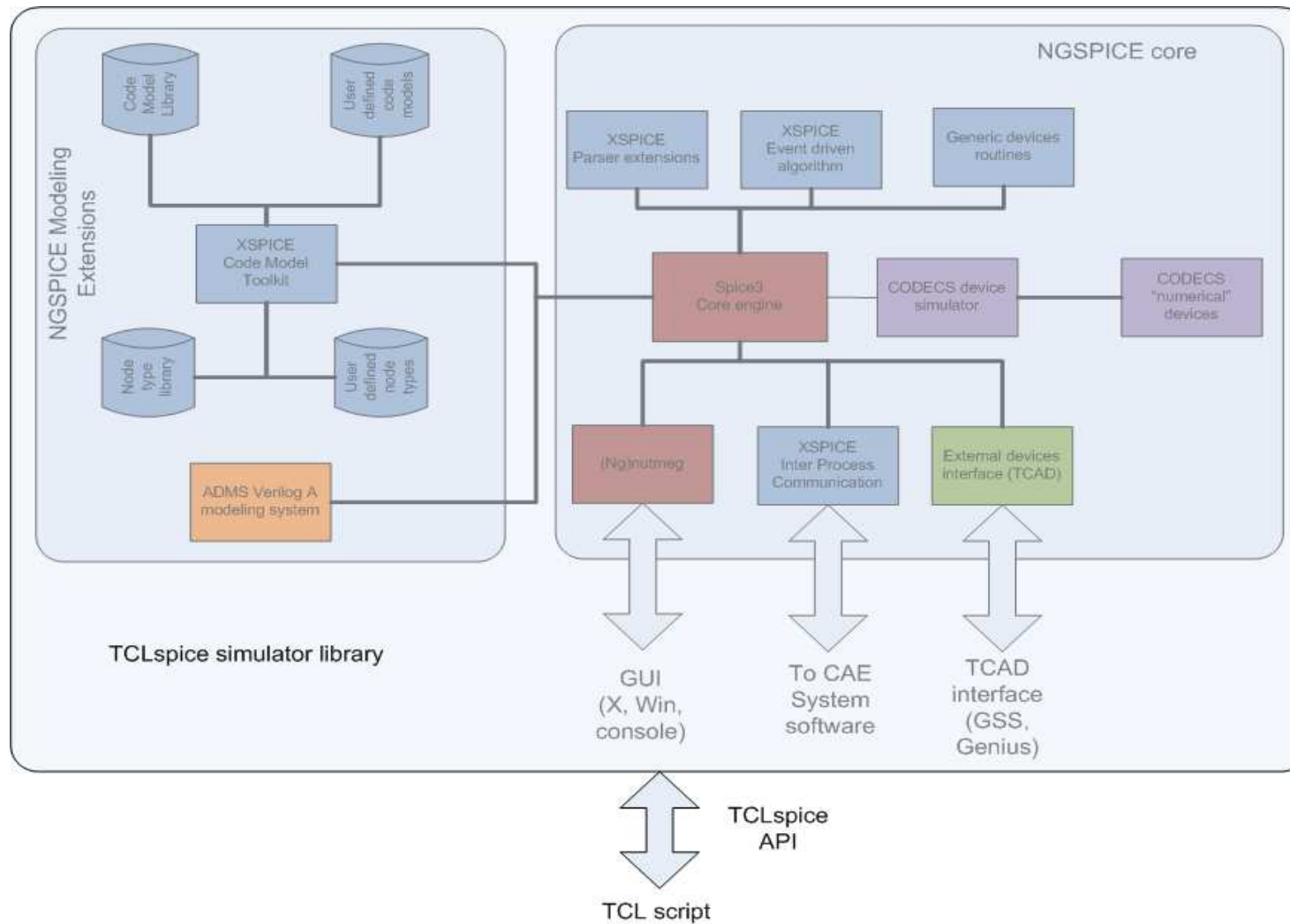
Re-thinking spice and ngspice



Source: IEEE Spectrum

- Spice (any spice-like simulator) alone is not able to verify such complex design in reasonable time.
- Spice and spice-like tools can be used to characterize part of a complex system (functional units of ICs) and provide higher-level, faster tools, the behavior of such units.

Ngspice as simulation kernel - TCLspice



TCLspice

- Tclspice started as an independent project by Stefan Jones and now is fully integrated into ngspice.
- TCLspice allows:
 - to simulate circuits and export data to post-processing applications
 - perform post synthesis of digital ASICs getting data from the TCL shell included in several digital design tools.
 - run simulations in an optimization loop.

TCLspice

- Example:
 - Extract the non-linear capacitance.
- In TCL we define the spice vector:
 - `spice::let Cim = real(mean(Vex#branch/(2*Pi*i*frequency*(V(5)-V(6)))))`
 - Run analyses at different voltages.
 - Plot the results.
 - Vector can be exported to another application driven by the same TCL script.

```

analyse-20070504-0.tcl
set pas [expr $v/ $v]

blt::vector create Ctmp
blt::vector create Cim
blt::vector create check

blt::vector create Vcmd
blt::graph .cimvd -title "Cim = f(Vd)"
blt::graph .checkvd -title "Rim = f(Vd)"

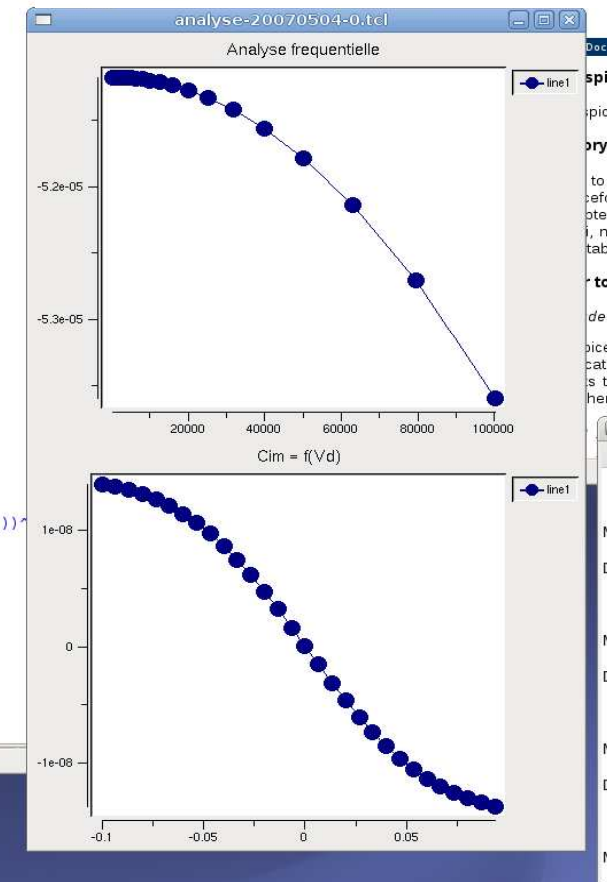
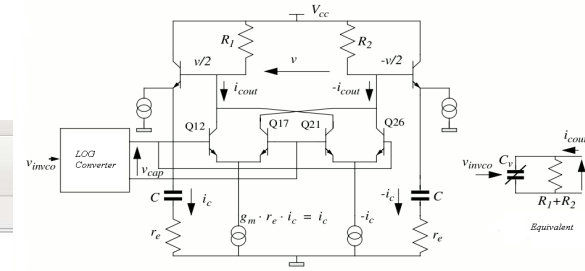
blt::vector create Iex
blt::vector create freq
blt::graph .freqanal -title "Analyse frequentielle"

set v [expr {$vmin + $n * $pas / 4}]
spice::alter vd = $v
spice::op
spice::ac dec 10 100 100k
spice::vectobl {Vex#branch} Iex
#spice::spicetobl {frequency} freq
spice::vectobl {frequency} freq
pack .freqanal
.freqanal element create line1 -xdata freq -ydata Iex

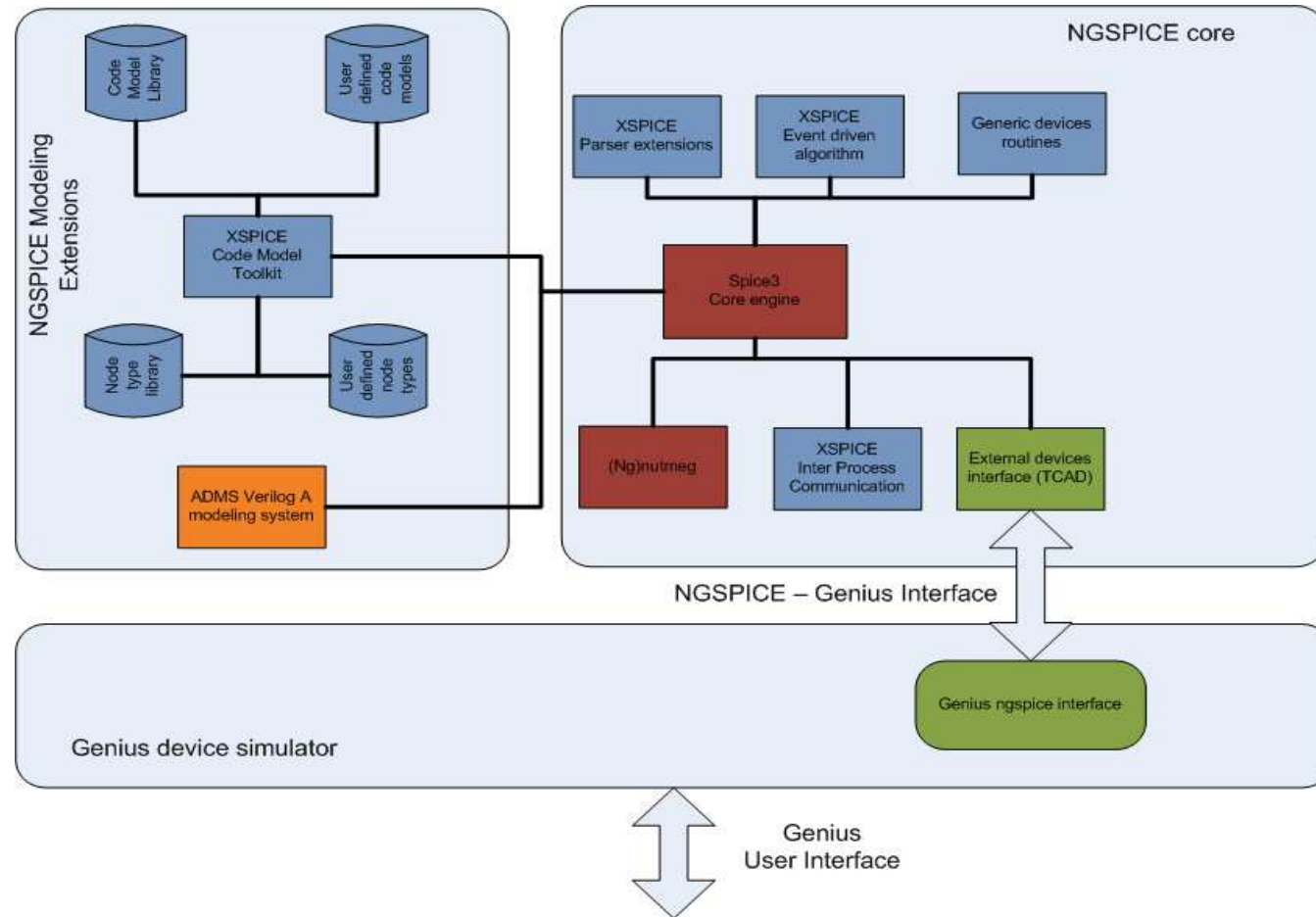
for {set i 0} {[expr $n - $i]} {incr i} {
    set v [expr {$vmin + $i * $pas}]
    spice::alter vd = $v
    spice::op
    spice::ac dec 10 100 100k

    spice::let Cim = real(mean(Vex#branch/(2*Pi*i*frequency*(V(5)-V(6))))
    spice::vectobl Cim Ctmp
    Cim append $Ctmp(0:end)
    #spice::let Cim = imag(mean(Vex#branch/(2*Pi*i*frequency*(V(5)-V(6))))
    spice::let err = real(mean(sqrt((Vex#branch-(2*Pi*i*frequency*Cim*V(5)-V(6))))
    #spice::print Vex#branch
    spice::vectobl err Ctmp
    check append $Ctmp(0:end)
    Vcmd append $v
}

pack .cimvd
.cimvd element create line1 -xdata Vcmd -ydata Cim
#pack .checkvd
#.checkvd element create line1 -xdata Vcmd -ydata check
    
```



Ngspice as simulation kernel – TCAD mixed mode



Genius TCAD simulator

- Genius is a TCAD tool designed by Cogenda (www.cogenda.com) and, partially released as open source under GPL V3:
 - 2D simulations (no 3D)
 - Serial execution (no parallel code)
 - Triangle mesh generator only
 - Basic set of materials
- Genius TCAD can simulate circuits containing TCAD-devices with ngspice:
 - ngspice is loaded as a shared library (DLL) inside Genius

Ngspice – Genius HOWTO

- Download from Cogenda web site:
 - Genius Device Simulator (GPL Source Code)
genius-20100411-open.tar.gz
 - NGSPICE (with patches for Genius)
ngspice-genius-20100220.tar.gz
- You will need also:
 - petsc-3.1-p6.tar.gz from
www.mcs.anl.gov/petsc/petsc-as/index.html
 - cgnslib_2.5-4.tar.gz from cgns.sourceforge.net/
 - VTK 5.4.2 from www.vtk.org

Ngspice – Genius HOWTO (2)

- Compile Genius according to the instructions in the INSTALL file.
- Ngspice must be compiled according to the following sequence:
 - run “autogen.sh” to create autoconf script needed to configure ngspice
 - run “build.sh”. This script configure ngspice to be used with Genius:
 - in particular enables the “ndev” device, the interface between the two simulators.
 - “make” ngspice
 - run “spicelib.sh”. This script creates a library from ngspice
 - cd into the “interface” directory and run “make”
 - copy “spice.so” in the lib directory of Genius installation

Ngspice - Genius HOWTO (3)

- The interface between Genius and NGSPICE is the “N” device:

Mixed Device/Circuit simulation of Diode

```
Vpp 2 0 0.3V sin(0.3 1.0 1MEGHZ)
```

```
N0 2=anode 3=cathode
```

```
R1 3 1 0.1
```

```
C0 1 0 10n
```

```
RL 1 0 1k
```

```
.option acct itl2=100
```

```
.tran 0.01us 10us
```

```
.print i(Vpp)
```

```
.plot tran i(Vpp)
```

```
.END
```

- The nodes of the “numerical device” must match the corresponding contacts in the TCAD file.

Ngspice - Genius HOWTO (3)

- The interface between Genius and NGSPICE is the “N” device:

```
# Genius example: PN Diode simulation mixed-type transient with NGSPICE
GLOBAL T=300 DopingScale=1e18 Z.Width=1
# load spice circuit netlist
CIRCUIT NETLIST=diode_ckt.cir
# Import CGNS file generated at first step
IMPORT CGNSFile=pn2d.cgns
# specify boundary condition.
boundary Type = OhmicContact ID = Anode
boundary Type = OhmicContact ID = Cathode
#Model Region=Silicon H.Mob=false Mob.force=ESimple EB.level=tl
METHOD Type=DDML1M NS=Basic LS=LU Damping=Potential maxit=30 #Fermi=On
SOLVE Type=Op
#SOLVE Type=DCSweep Vscan=Vpp Vstart=0.0 Vstep=0.1 Vstop=2.5 out.prefix=diode_mix
SOLVE Type=tran tstart=0.0 tstep=0.001e-6 tstepmax=0.01e-6 tstop=6e-6 out.prefix=diode_mix
EXPORT VTKFILE=diode.vtu
```

- The nodes of the “numerical device” must match the corresponding contacts in the TCAD file.

Mixed level simulation – CIDER Device simulator

- The idea of mixing TCAD and spice simulations is not new.
- 1994 CIDER 1b1:
 - Couples spice3f.2 to a C-based device simulator DSIM.
 - Poisson and continuity equation solver in 1D and 2D compatible with Stanford's PISCES.
 - Netlists can contains compact models and “numerical” models.

Mixed level simulation – CIDER Device simulator

Model	Description
numd	1D junction diode model
numd2	2D junction diode model
numbjt	1D bjt model
numbjt2	2D bjt model
numos	2D mosfet model

- A fixed number of simulable devices is defined.
- Only 1D and 2D device models support.

Mixed level simulation – CIDER Device simulator

TWO-DIMENSIONAL PIN-DIODE CIRCUIT

```
VIN 1 0 0.0v (PWL 0ns 0.8v 1ns -50.0v)
L1 1 2 0.5uH
VD 2 3 0.0v
D1 3 0 M_PIN AREA=200 IC.FILE="OP.0.d1"
VRC 2 4 0.0v
R1 4 5 100
C1 5 0 1.0nF
```

```
.MODEL M_PIN NUMD LEVEL=2
```

```
+ options defw=1000u
```

```
+ x.mesh n=1 l=0.0
```

```
+ x.mesh n=2 l=0.2
```

```
+ x.mesh n=4 l=0.4
```

```
+ x.mesh n=8 l=0.6
```

```
+ x.mesh n=13 l=1.0
```

```
+ y.mesh n=1 l=0.0
```

```
+ y.mesh n=9 l=4.0
```

```
+ y.mesh n=24 l=10.0
```

```
+ y.mesh n=29 l=15.0
```

```
+ y.mesh n=34 l=20.0
```

```
+ domain num=1 material=1
```

```
+ material num=1 silicon tn=20ns tp=20ns
```

```
+ electrode num=1 x.l=0.6 x.h=1.0 y.h=0.0
```

```
+ electrode num=2 y.l=20.0
```

```
+ doping gauss p.type conc=1.0e20
```

```
+ char.len=1.076 x.l=0.75 x.h=1.1 y.h=0.0
```

```
+ lat.rotate ratio=0.1
```

```
+ doping unif n.type conc=1.0e14
```

```
+ doping gauss n.type conc=1.0e20
```

```
+ char.len=1.614 x.l=-0.1 x.h=1.1 y.l=20.0
```

```
+ models bgn srh auger conctau concmob fieldmob
```

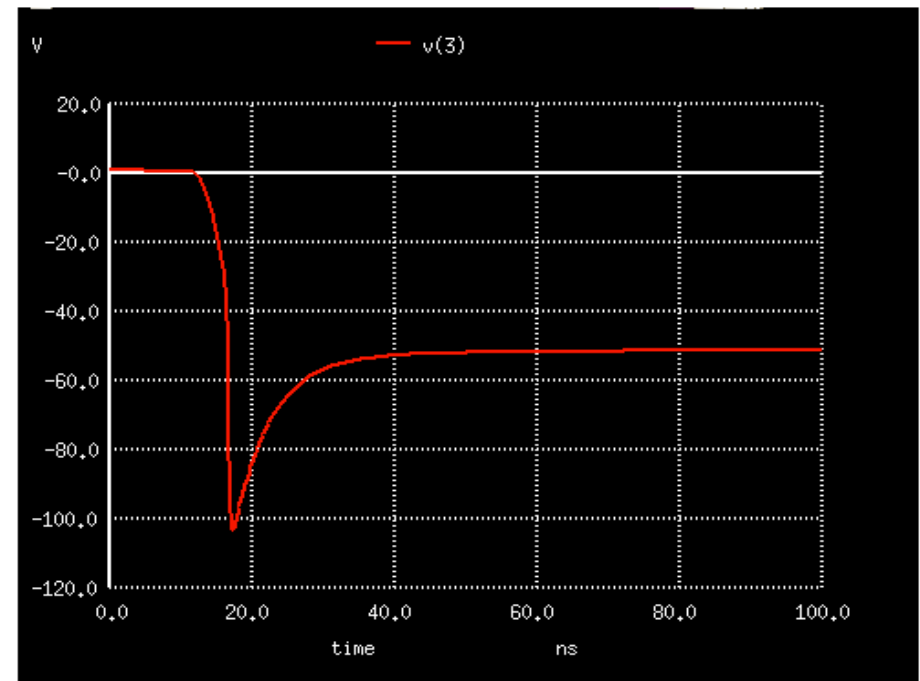
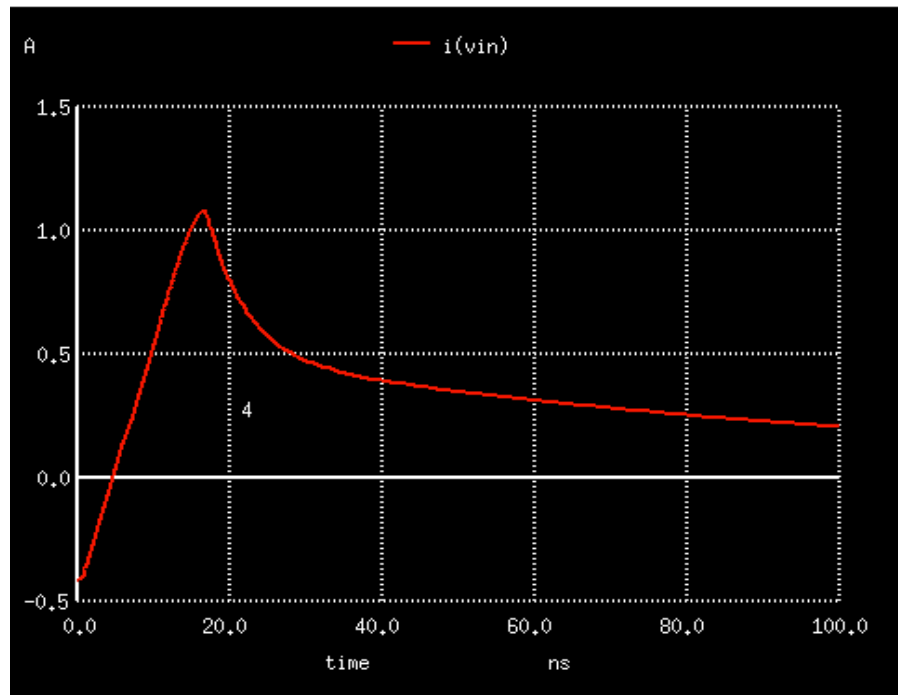
```
.OPTION ACCT BYPASS=1
```

```
.TRAN 1NS 100NS
```

```
.PRINT TRAN v(3) I(VIN)
```

```
.END
```

CIDER Simulation – PIN diode transient



Applications of

NGSPICE AT DIET (UNIVERSITÀ LA SAPIENZA)

20 years later...

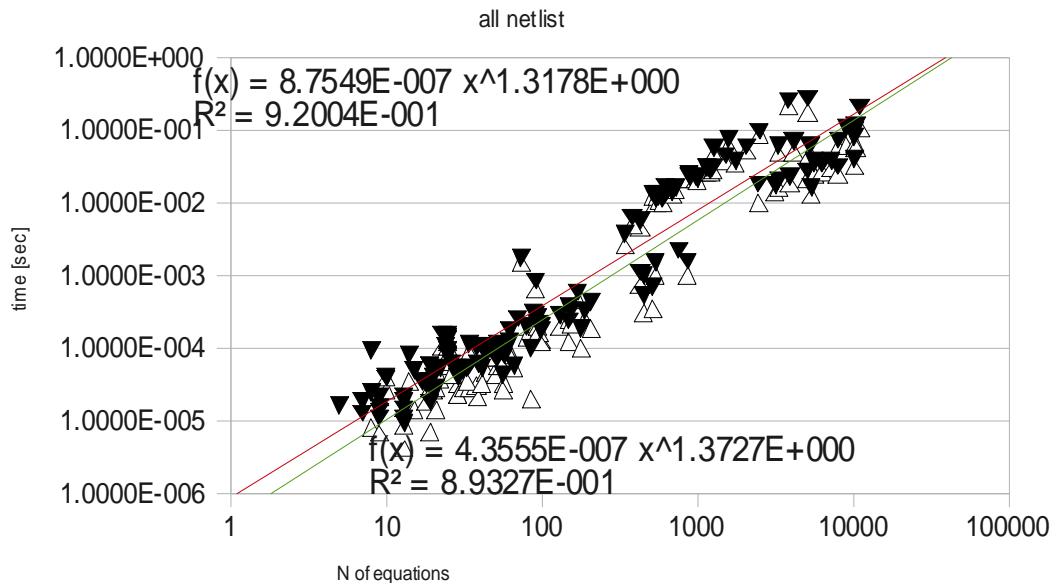
PROFILING NGSPICE

Profiling ngspice

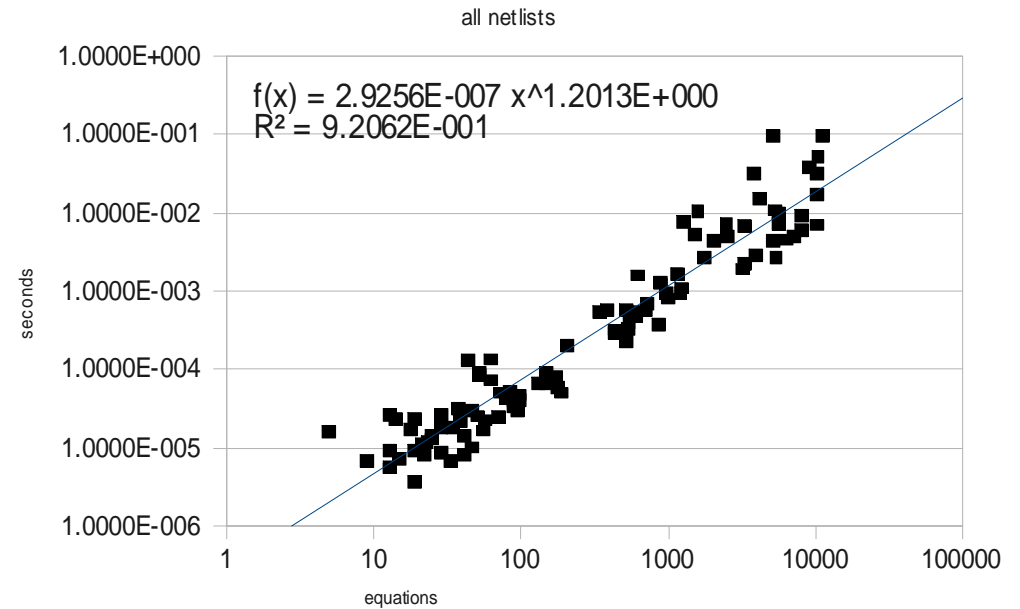
- Latest profile data dates back to Quarles thesis:
 - We need newer data for ngspice and newer CPUs.
 - We need cache usage data.
- Tools :
 - Gprof code profiler
 - Cachegrind (Valgrind) cache profiler
- Test suite (expanding):
 - Quarles Bench:
 - 68 netlists, BJT, Mos1, Mos2, Mos3.
 - Equations: from 4 a 451.
 - Devices da 2 a 1218
 - CircuitSim90:
 - 37 netlists, BJT, Mos2, Mos3.
 - Equations: from 13 to 5093.
 - Devices: from 8 to 5091.
 - Iscas85/89:
 - 45 netlists, BSIM3v1.
 - Equations: 73 to 11029,
 - Devices: 289 a 25116.

Transient analysis performances

total and load time/total iteration vs Circuit Equations

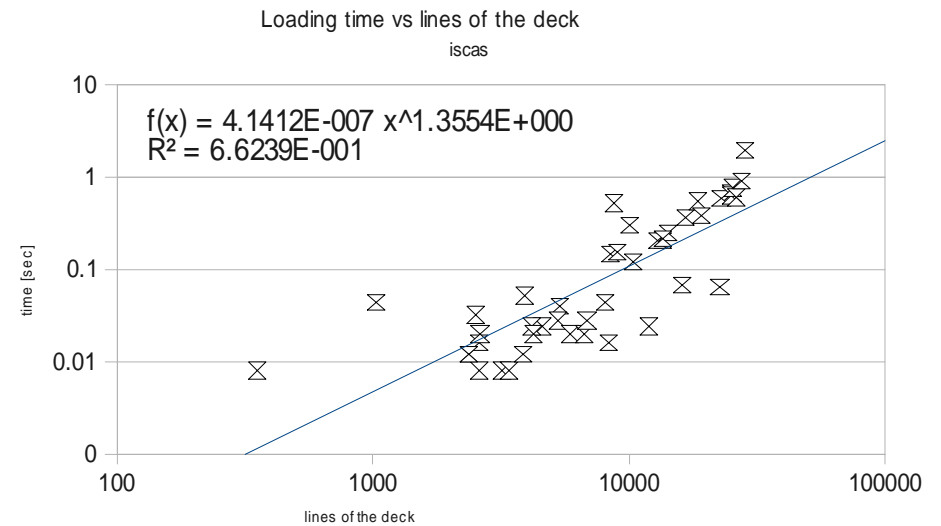
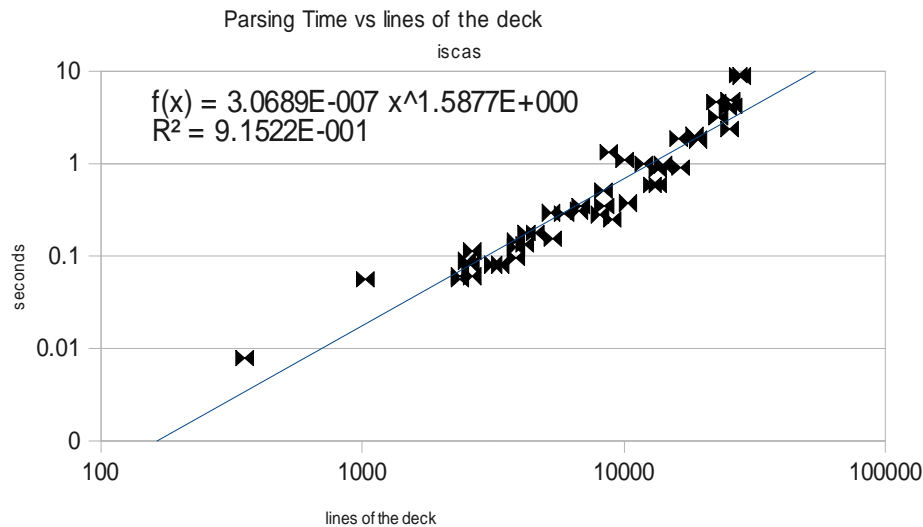


Solve time/transient iteration vs Circuit equations



- Performances over number of circuit equations:
 - Total analysis time $8.7549E-007 * N^{1.3178}$
 - Total load time $4.3555E-007 * N^{1.3727}$
 - Total solve time $2.9256E-007 * N^{1.2013}$

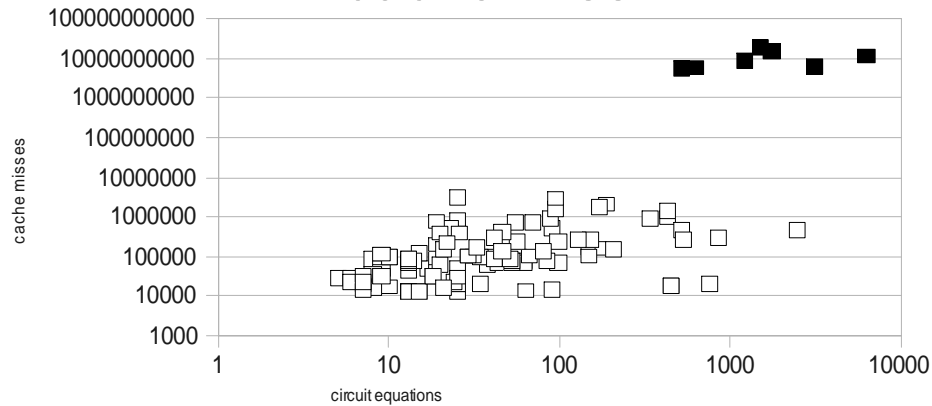
Netlist parsing and loading time



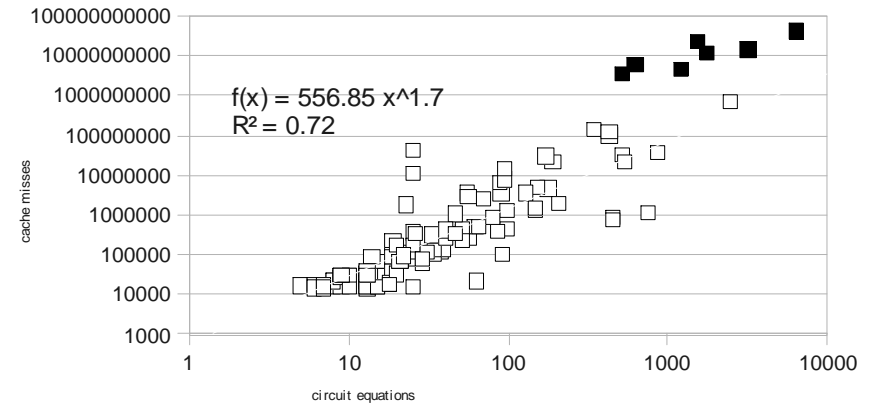
- Performances over number of deck lines:
 - Parsing time: $3.0689E-007 * L^{1.5877}$
 - Loading time: $4.1412E-007 * L^{1.3554}$

Cache misses over circuit equations

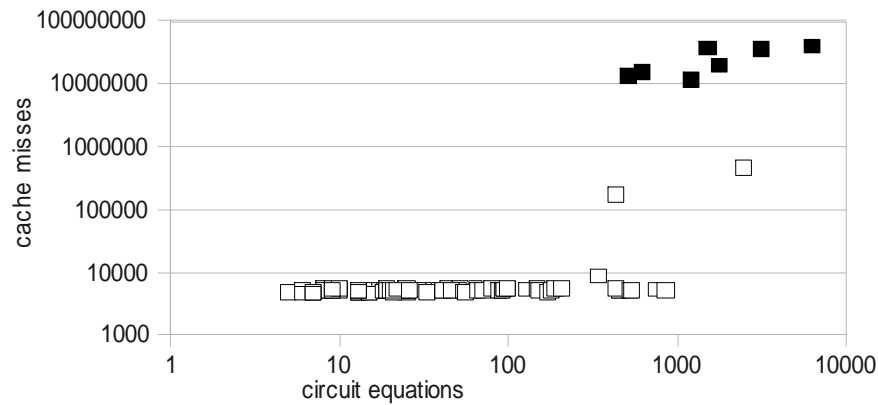
I1 cache miss



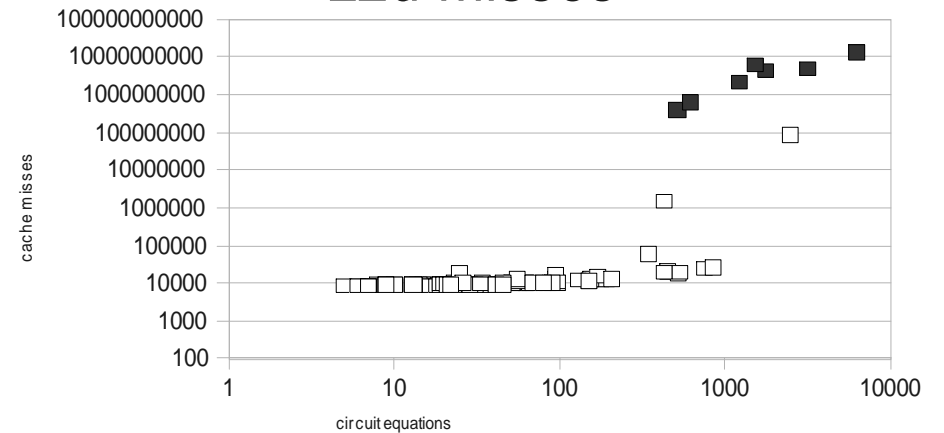
D1 cache miss



L2i misses



L2d misses



Cache miss analysis

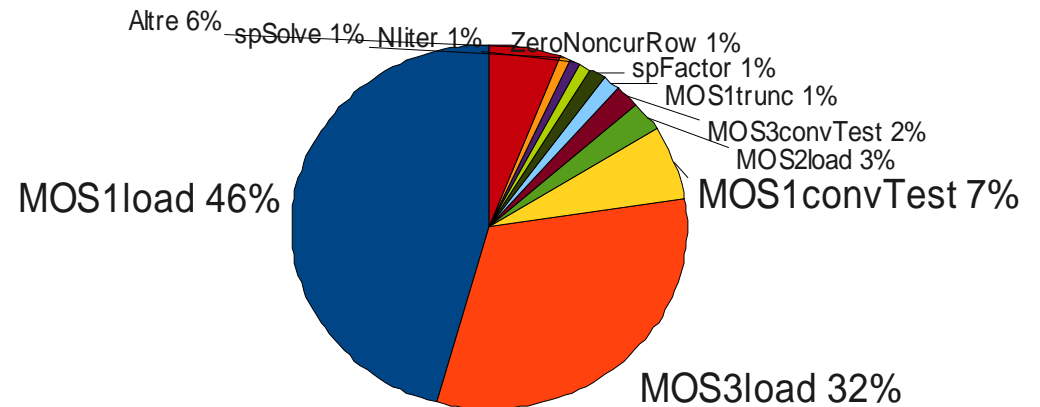
- Quarles Bench benchmarks:

- Level 1 instruction cache
 - DEVload (Mos1,2,3):
 - 64% Cache miss
 - 0%,0.02%, 0.01% Cache miss rate
 - Level 1 data cache
 - DEVload (Mos1,2,3)
 - 80% Cache miss
 - 0.01% Cache miss rate

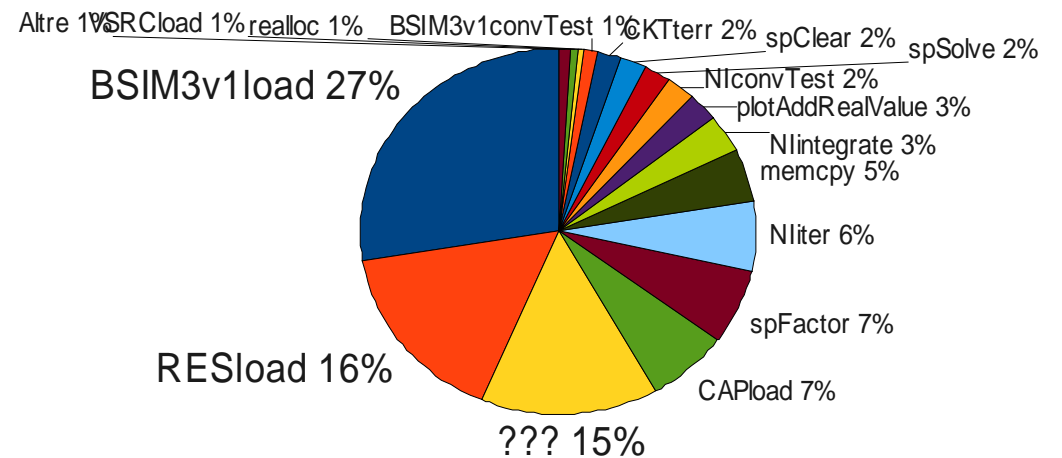
- ISCAS benchmarks:

- Level 1 instruction cache
- BSIM3v1load
 - 96% Cache miss
 - 2% Cache miss rate
- Level 1 data cache
- SpFactor
 - 25% Cache miss
 - 5% Cache miss rate

Quarles Bench



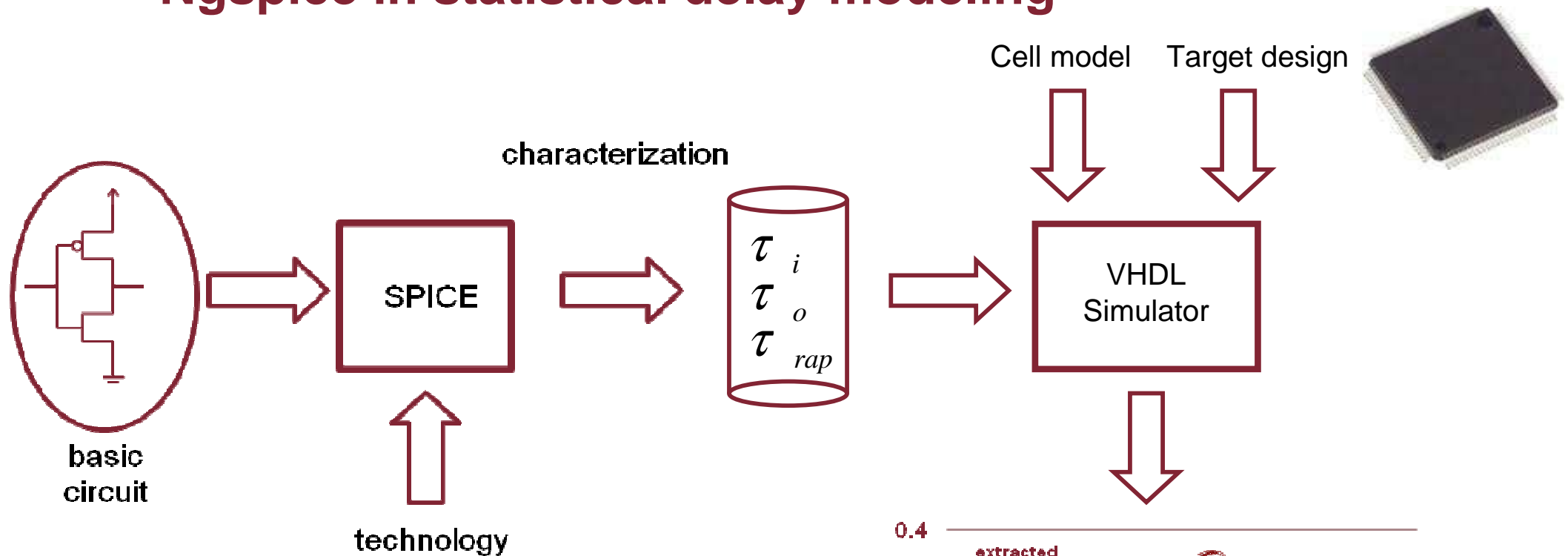
Iscas



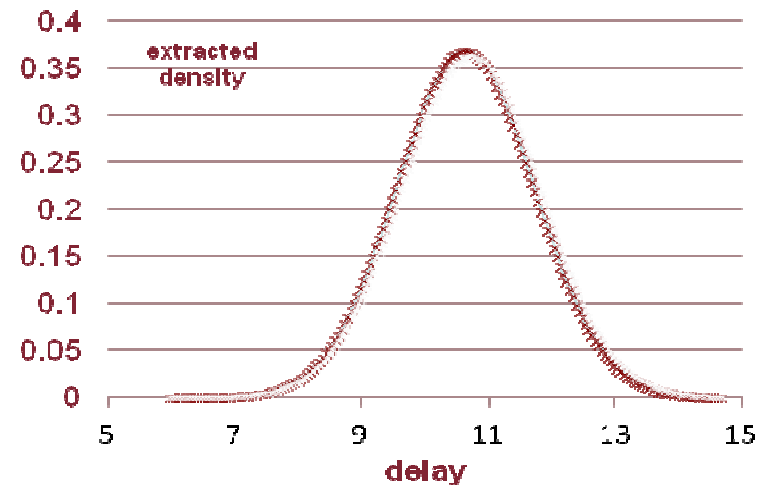
Nano CMOS standard cells library

STATISTICAL DELAY MODELING

Ngspice in statistical delay modeling

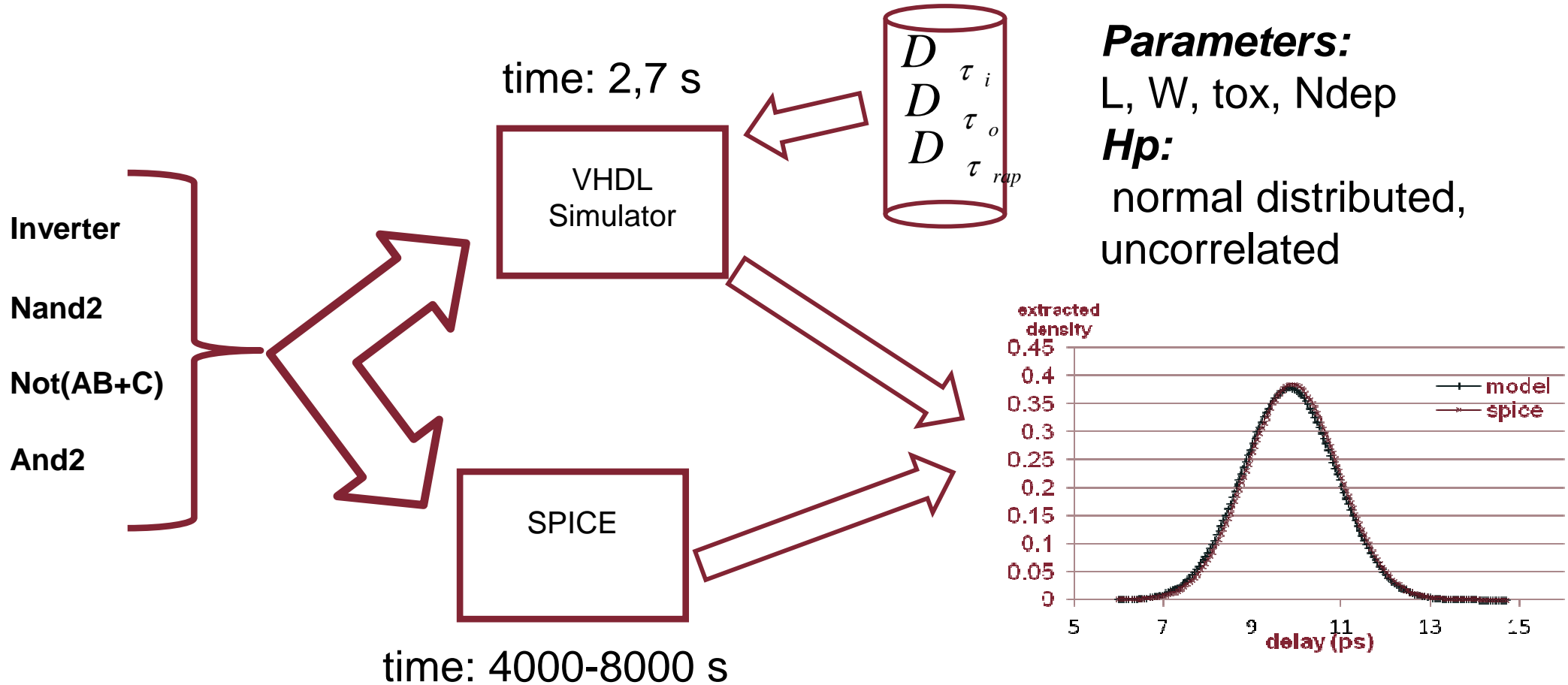


Ngspice is used to characterize the single cells under all possible operating conditions to extract delay data that can be fed to a VHDL simulator for full chip design verification.



Ngspice in statistical delay modeling

Comparison between spice and VHDL simulation time over 10.000 runs

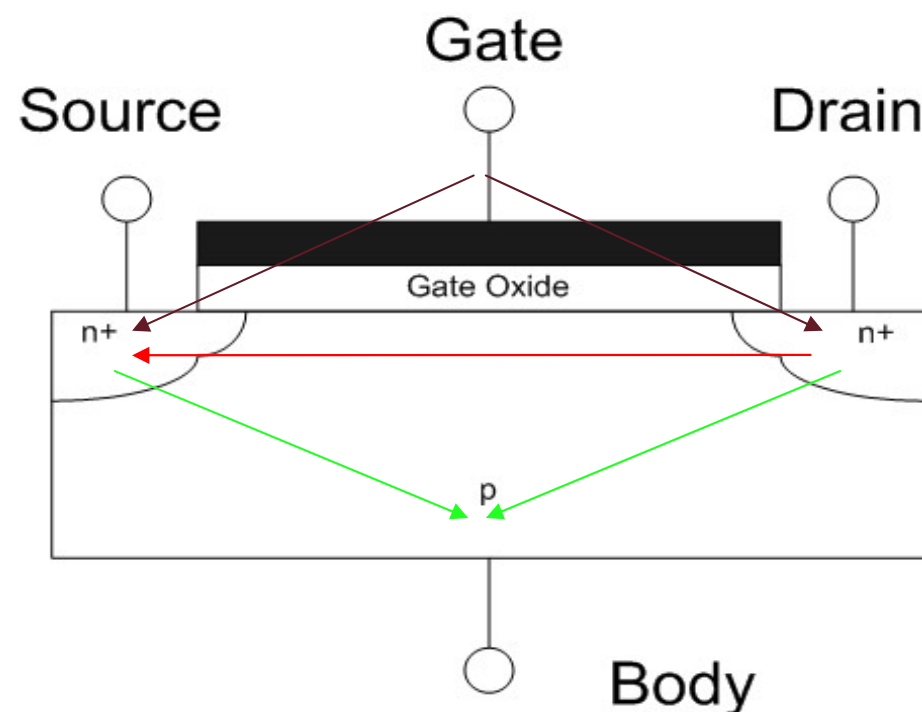


Simplified MOSFET models for evaluation of

STATIC LEAKAGE IN LOGIC CELLS

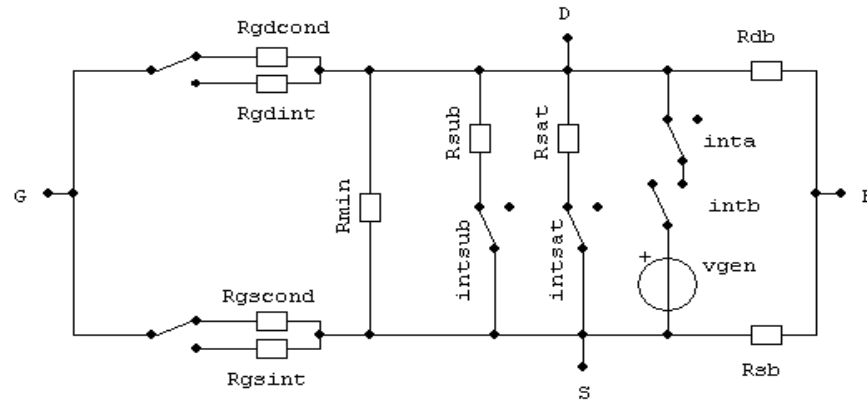
Leakage currents in a MOS transistor

- Static leakage is a great concern in scaled technologies.
- We consider the following contributions to leakage:
 - Gate Leakage.
 - Sub-Threshold leakage.
 - BTBT (Band-to-Band-Tunneling).

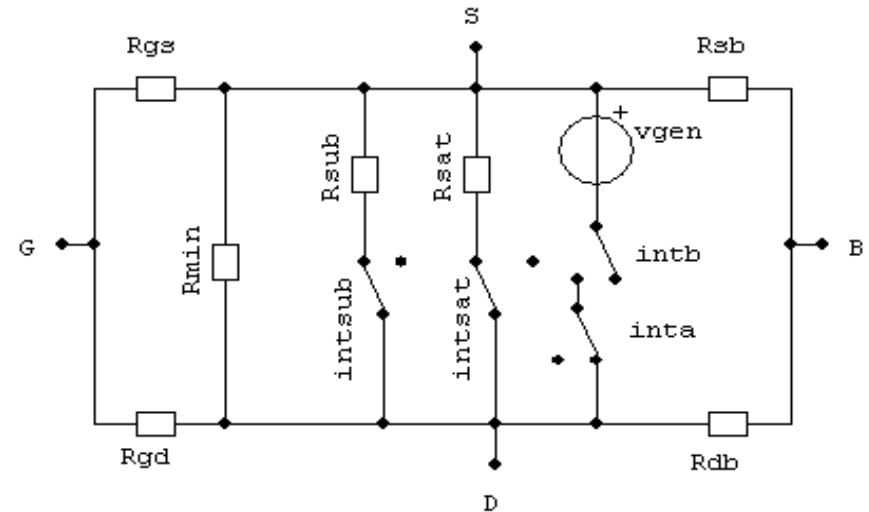


Linear MOSFET model for leakage estimation

nMOS

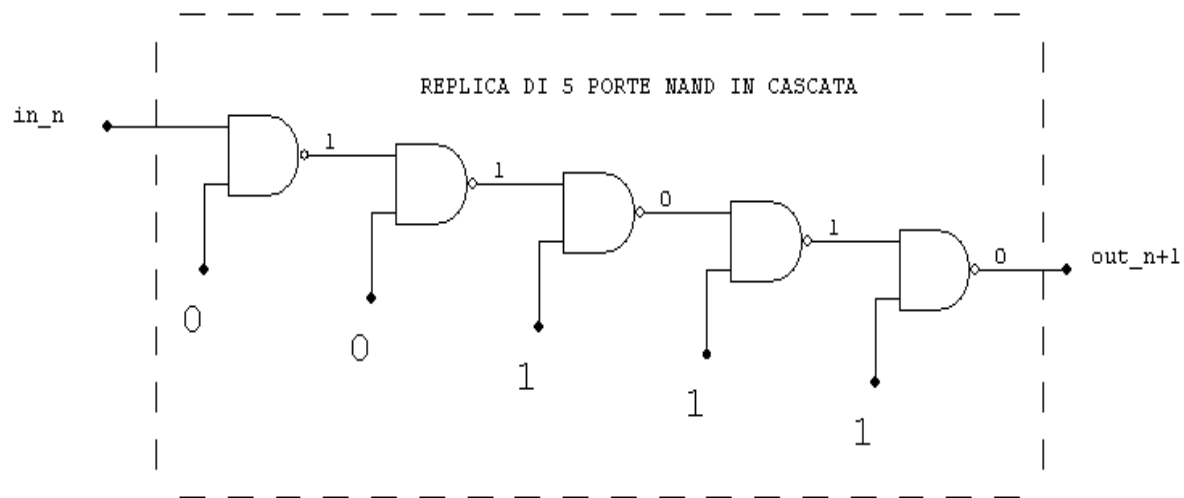
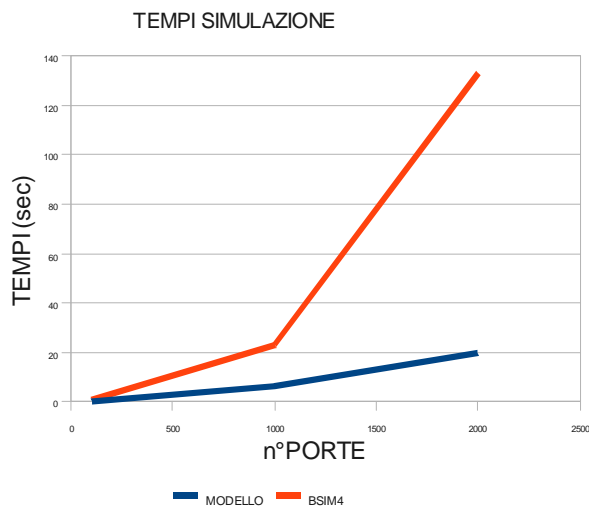
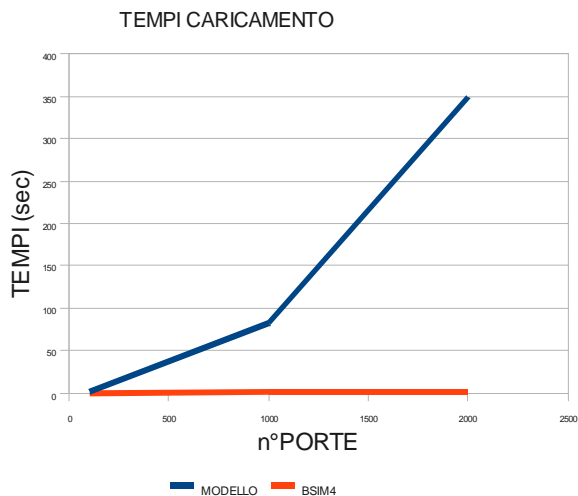


pMOS



- Replace each transistor in the circuit with a simplified model consisting only of linear conductances, switches and voltage generators.
- Ngspice is used to perform parameter extraction for the simplified model.

Simulation times for cascaded gates

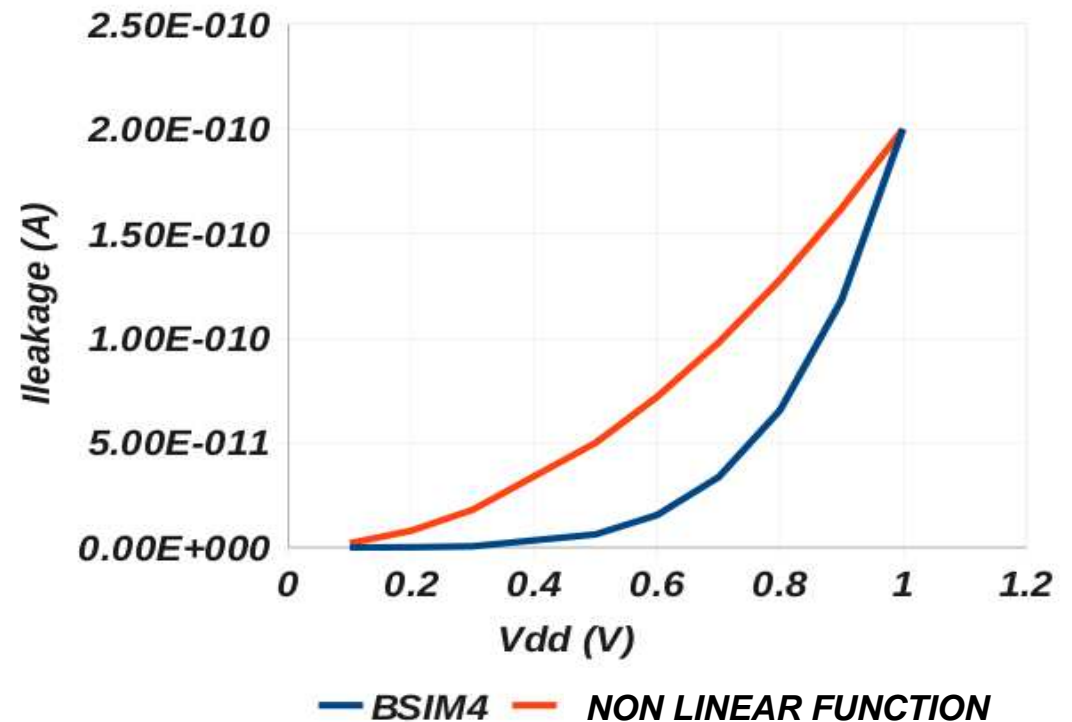
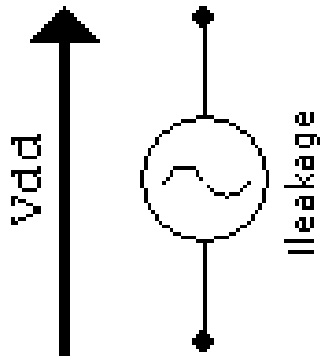


<i>Gates</i>	<i>100</i>		<i>1000</i>	
<i>Time</i>	<i>Model</i>	<i>BSIM4</i>	<i>Model</i>	<i>BSIM4</i>
Loading (sec)	0.6	0.13	82	0.48
Simulation (sec)	0.22	0.74	6	22.88
Total(sec)	0.82	0.87	88	23

Non-linear MOSFET model for leakage estimation

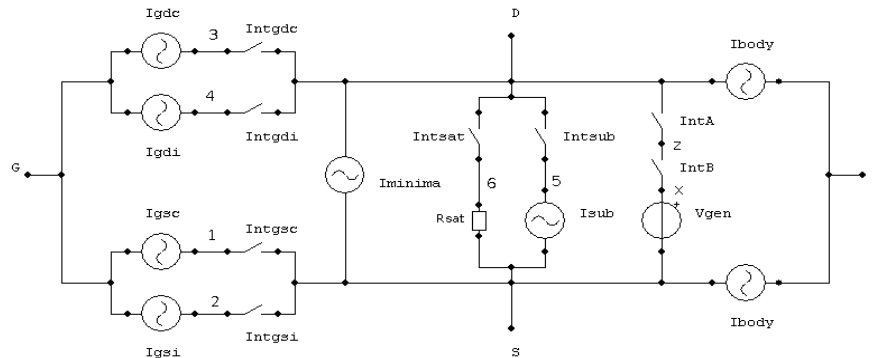
Each current in the non linear model is expressed as a non linear current source:

$$I_{\text{leakage}} = \alpha * V_{\text{dd}}^{\beta}$$

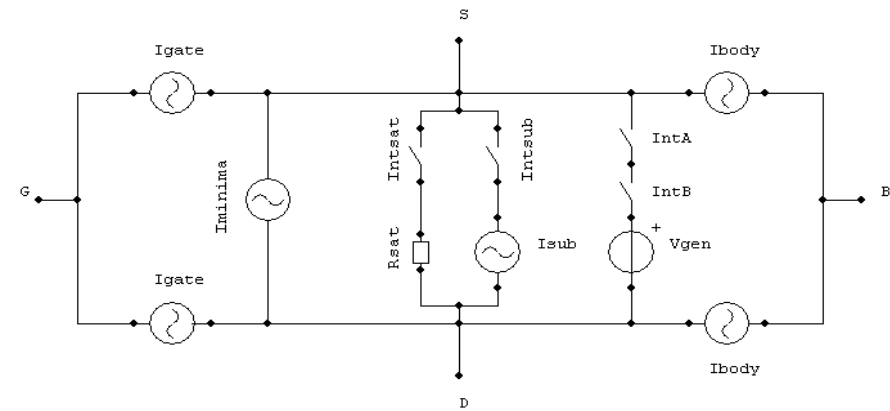


Non-linear MOSFET models for leakage estimation

nMOS



pMOS



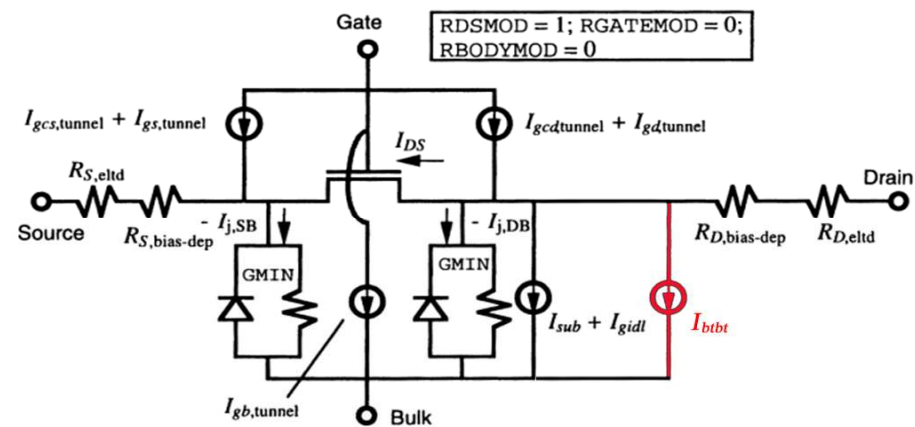
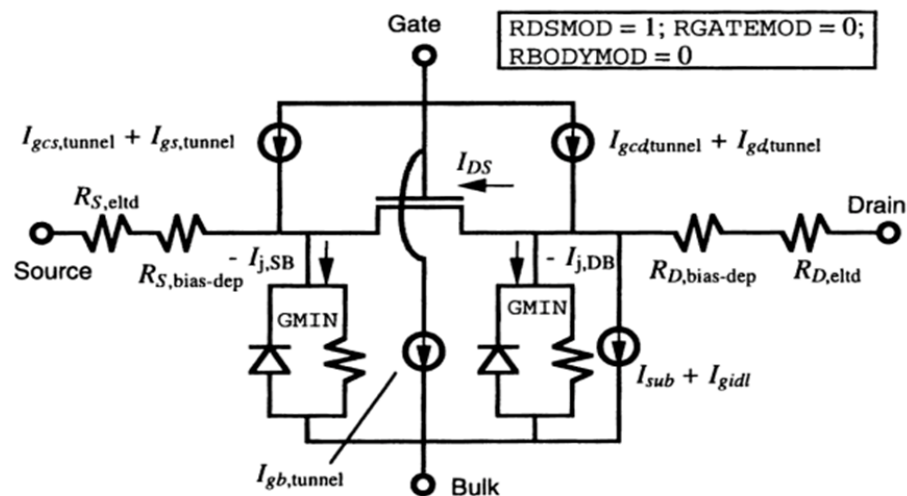
- Each transistor in the netlist is replaced by a “non-linear switched current source model”
- This model is under development.

First integration of

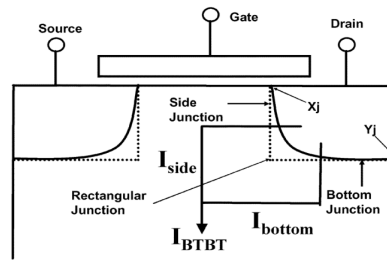
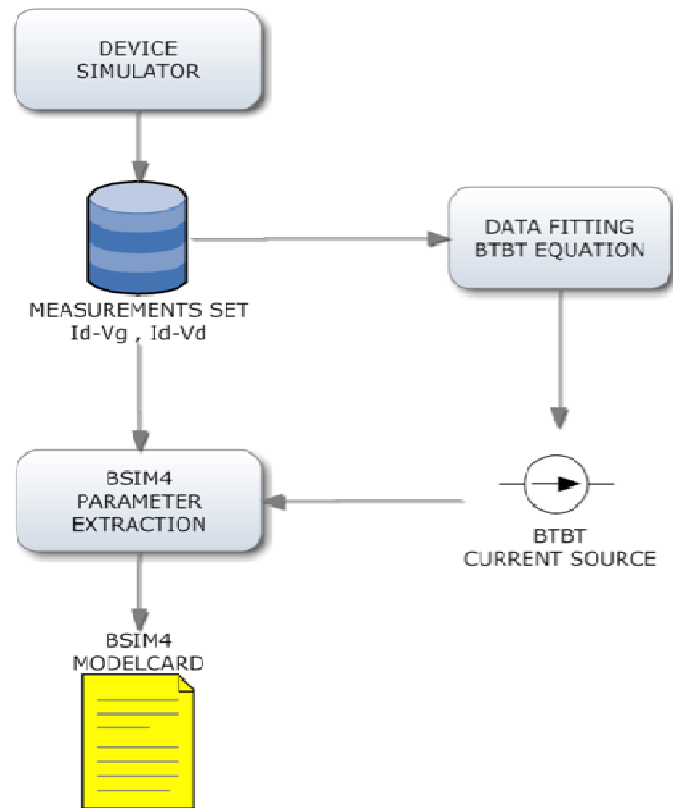
BTBT CURRENT GENERATOR IN BSIM4

BTBT leakage in BSIM4

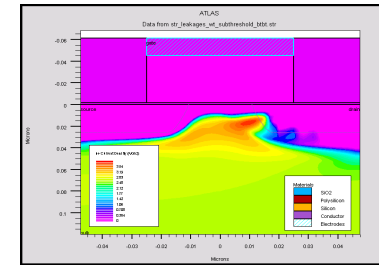
- A model of leakage current has been introduced in BSIM4 DC model.
- Ngspice has been used inside the parameter extraction loop.



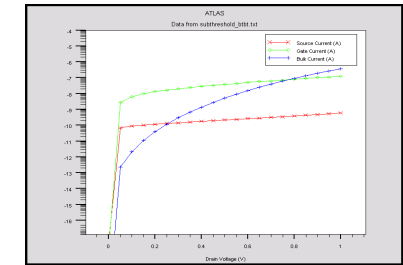
Workflow for BTBT source extraction



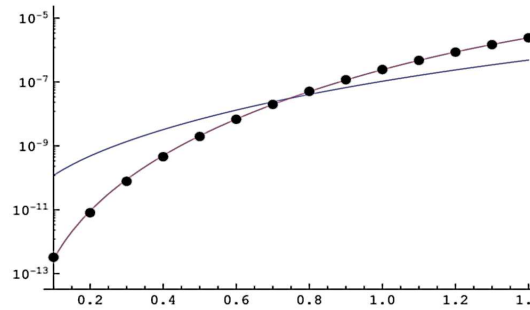
Rectangular junction approximation



BTBT current density



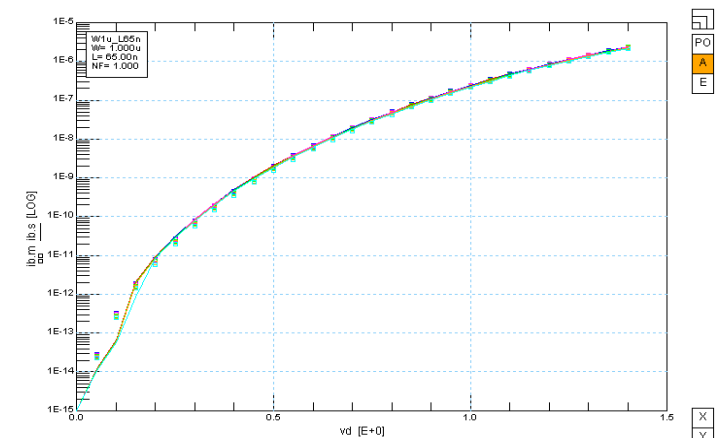
Leakage currents



Comparison between the analytical model and TCAD simulations

$$I_{BTBT\text{drain}} = w_{\text{eff}} |Y_j| J_{\text{side}} + w_{\text{eff}} |x_{\text{max}} - X_j| J_{\text{bottom}}$$

$$I_{BTBT\text{corrected}} = I_{BTBT\text{drain}} \cdot a_0 \left[\exp\left(\frac{\alpha_{btbt} V_{\text{app}}^{1+m}}{\beta_{btbt} + \gamma_{btbt} V_{\text{app}}^m}\right) - 1 \right]$$



BTBT

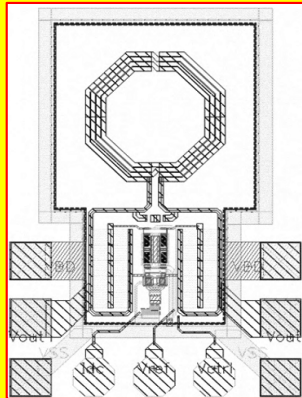
Hypotesis for

FUTURE DEVELOPMENTS

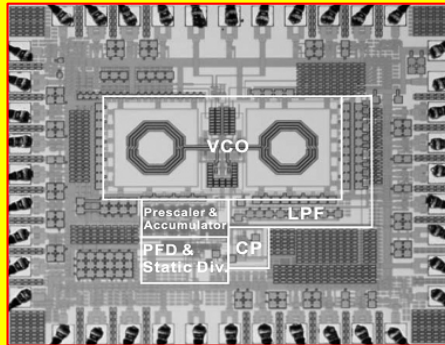
Implementing PSS in NGS spice

Periodic Steady State (PSS) is a dedicated RF large-signal circuit analysis

Autonomous Systems

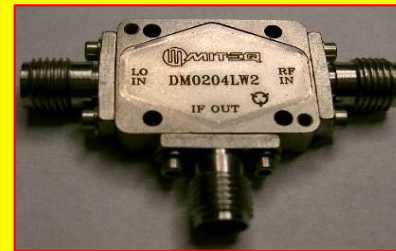


Oscillators
and VCOs

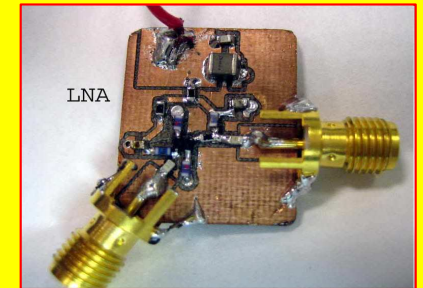


PLLs

Non-Autonomous Systems



Mixers



LNAs

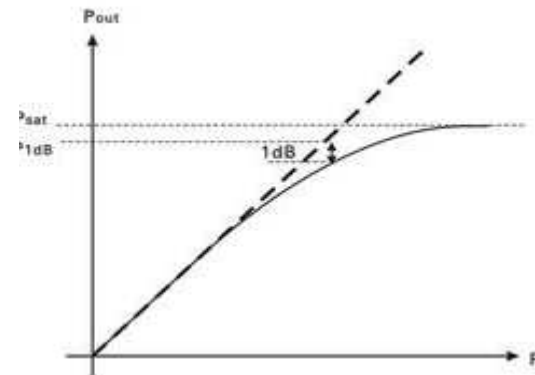


PAs

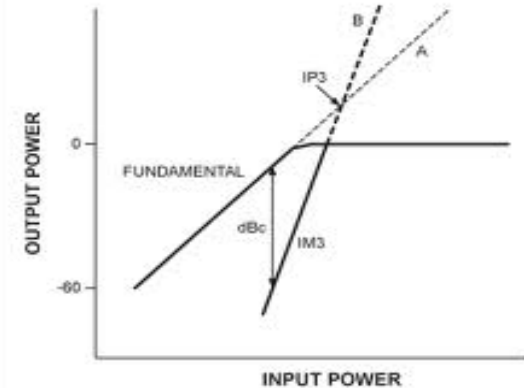
Implementing PSS in NGSpice

PSS is the most accurate tool in simulated characterization of RF systems.

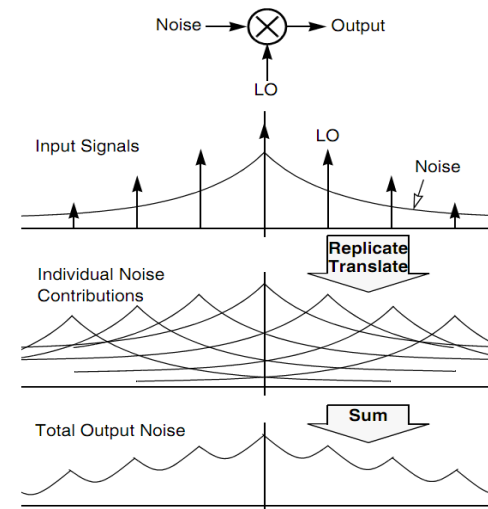
- compression point, IP3 and other non linear harmonic content evaluations
- cyclo-stationary noise evaluation as folding in mixers and phase noise in oscillators



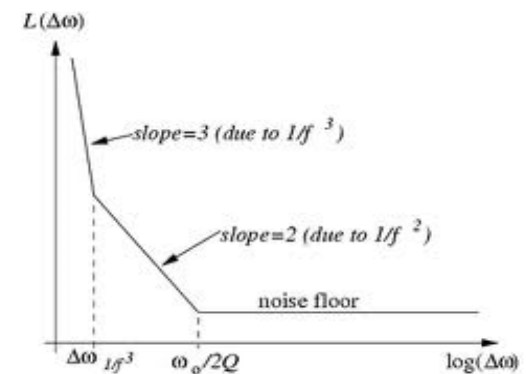
1dB compression point



IP3 evaluation



mixer noise



oscillator phase noise

Integration of HomSpice and HomSSPICE

- The integration of homotopy-based simulators HomSpice and HomSSpice is planned.
- Both simulator are based on SPICE3C1 and HOMPACT library (fixpdf, fixnf, fixqf).
- Homotopy methods are robust and accurate.
- DC and periodic steady-state analysis is critical for simulation of lcs.
- Solvers based on Newton methods often exhibit convergence problems when used to simulate highly nonlinear circuits and circuits with multiple dc operating points.
- Steady-state based on transient analysis require simulation over long transient periods.

Homotopy methods

- Original problem:

$$F(x) = 0$$

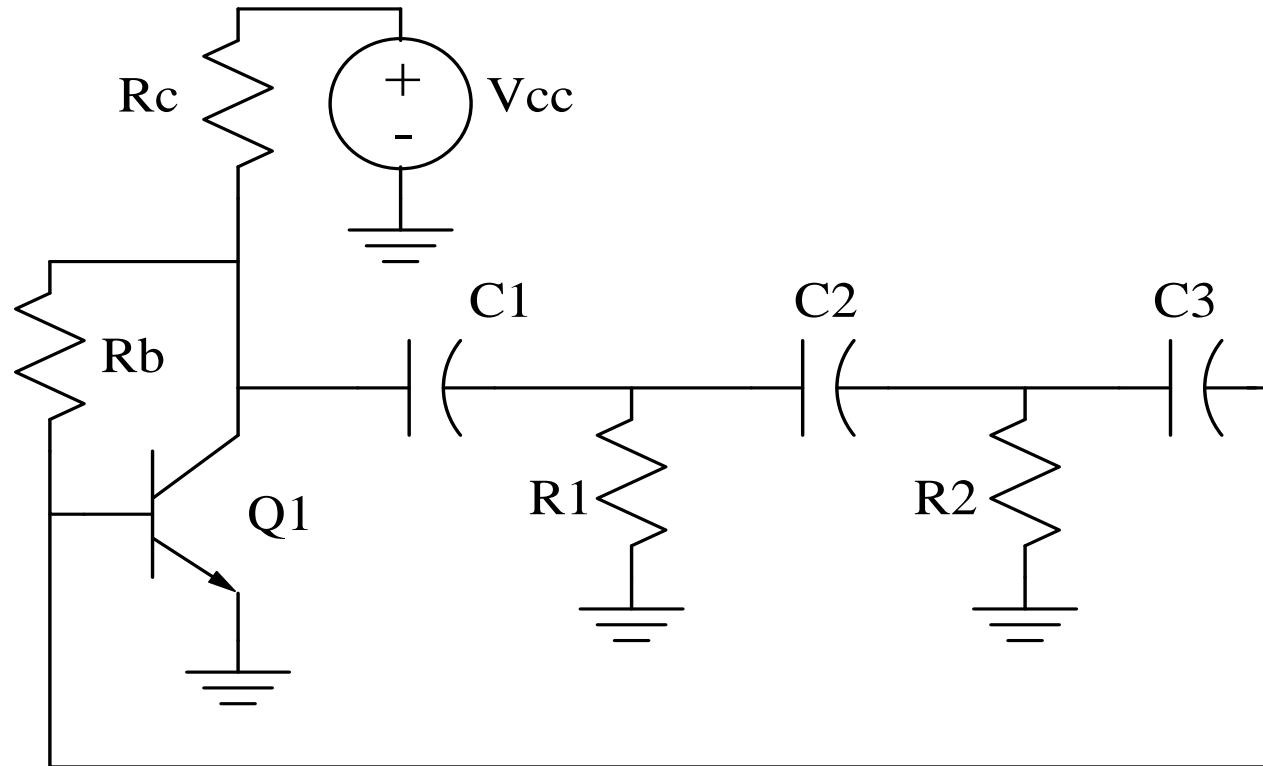
- Homotopy:

$$\rho_a(x, \lambda) = \lambda F(x) + (1 - \lambda)(x - a)$$

- Arc-length (artificial parameter) homotopy:

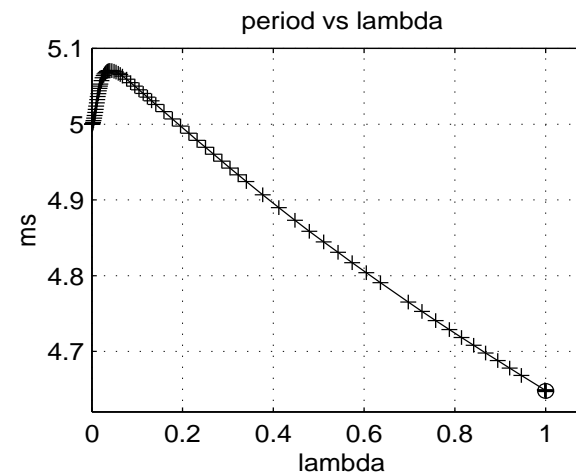
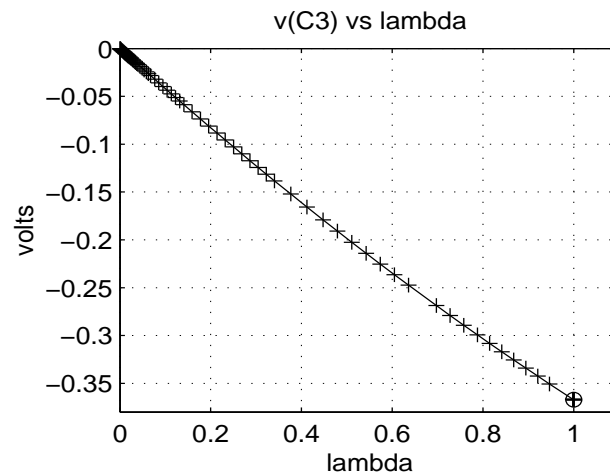
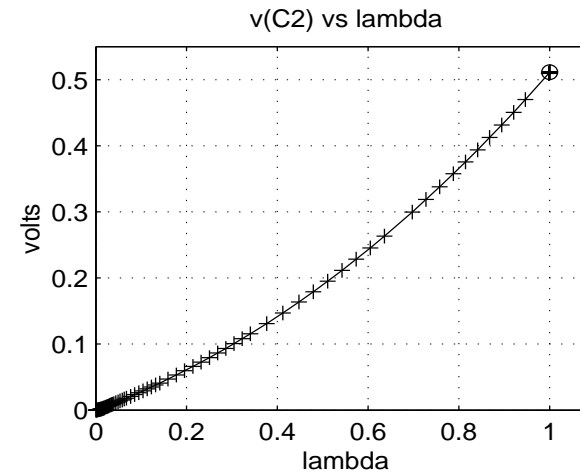
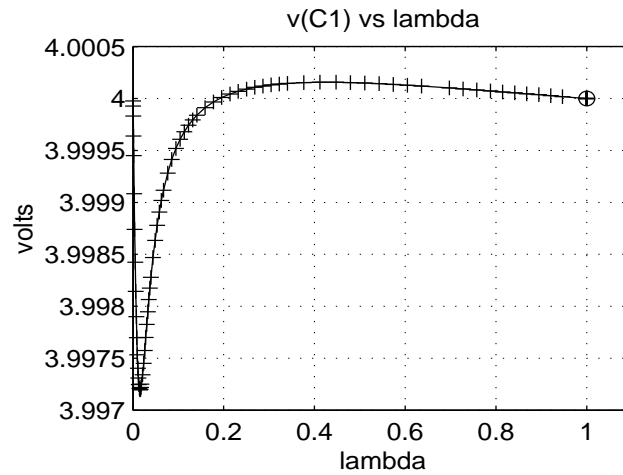
$$\rho_a(x(s), \lambda(s)) = \lambda(s)F(x) + (1 - \lambda(s))(x - a)$$

Phase shift oscillator



$R_c = 3.3 \text{ K}\Omega$, $R_b = 470 \text{ K}\Omega$, $R_1 = 7 \text{ K}\Omega$, $R_2 = 5 \text{ K}\Omega$, $C_1 = C_2 = C_3 = 47 \text{ nF}$, and $V_{cc} = 10 \text{ V}$

Phase shift oscillator



Conclusions

- Spice (and Ngspice), after 40 years are still invaluable tools for circuit design, analysis and verification.
- Ngspice is a versatile platform integrable in higher-level applications to provide accurate analog simulation
- Applications of Ngspice at Department of Information Engineering, Electronics and Telecommunications (DIET) have been reported.
- Future development have been presented
- Visit ***www.ngspice.org***

Q&A
